

Informe Laboratorio 4

Sección 1

Tobías Guerrero Cheuquepán
e-mail: tobias.guerrero_c@mail.udp.cl

Noviembre de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo (Parte 1)	3
2.1. Detecta el cifrado utilizado por el informante	3
2.2. Logra que el script solo se gatille en el sitio usado por el informante	4
2.3. Define función que obtiene automáticamente el password del documento	4
2.4. Muestra la llave por consola	5
3. Desarrollo (Parte 2)	6
3.1. Reconoce automáticamente la cantidad de mensajes cifrados	6
3.2. Muestra la cantidad de mensajes por consola	6
4. Desarrollo (Parte 3)	7
4.1. Importa la librería cryptoJS	7
4.2. Utiliza SRI en la librería CryptoJS	7
4.3. Logra descifrar uno de los mensajes	8
4.4. Imprime todos los mensajes por consola	9
4.5. Muestra los mensajes en texto plano en el sitio web	10
4.6. El script logra funcionar con otro texto y otra cantidad de mensajes	11
4.7. Indica url al código .js implementado para su validación	13

1. Descripción de actividades

Para este laboratorio, deberá utilizar Tampermonkey y la librería CryptoJS (con SRI) para lograr obtener los mensajes que le está comunicando su informante. En esta ocasión, su informante fue más osado y se comunicó con usted a través de un sitio web abierto a todo el público <https://cripto.tiiny.site/>.

Sólo un ojo entrenado como el suyo logrará descifrar cuál es el algoritmo de cifrado utilizado y cuál es la contraseña utilizada para lograr obtener la información que está oculta.

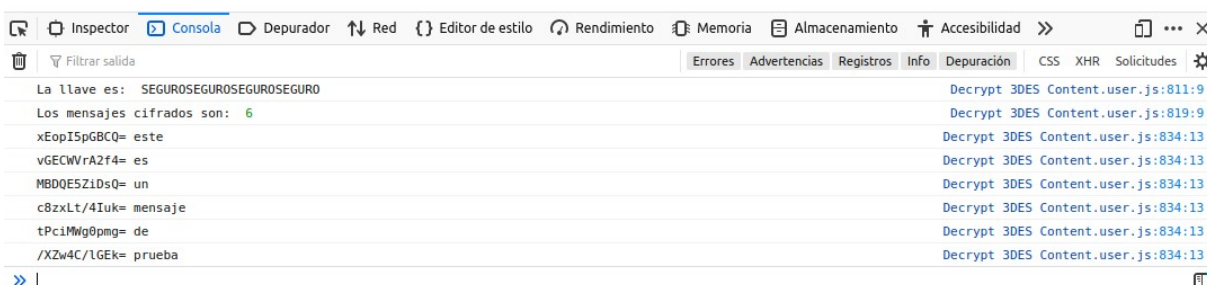
1. Desarrolle un plugin para tampermonkey que permita obtener la llave para el descifrado de los mensajes ocultos en la página web. La llave debe ser impresa por la consola de su navegador al momento de cargar el sitio web. Utilizar la siguiente estructura:
 - La llave es: KEY
2. En el mismo plugin, se debe detectar el patrón que permite identificar la cantidad de mensajes cifrados. Debe imprimir por la consola la cantidad de mensajes cifrados. Utilizar la siguiente estructura: Los mensajes cifrados son: NUMBER
3. En el mismo plugin debe obtener cada mensaje cifrado y descifrarlo. Ambos mensajes deben ser informados por la consola (cifrado espacio descifrado) y además cada mensaje en texto plano debe ser impreso en la página web.

El script desarrollado debe ser capaz de obtener toda la información del sitio web (llave, cantidad de mensajes, mensajes cifrados) sin ningún valor forzado. Para verificar el correcto funcionamiento de su script se utilizará un sitio web con otro texto y una cantidad distinta de mensajes cifrados. Deberá indicar la url donde se podrá descargar su script.

Un ejemplo de lo que se debe visualizar en la consola, al ejecutar automáticamente el script, es lo siguiente:

Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica. Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica. Sin el conocimiento de información secreta, el criptoanálisis se dedica al estudio de sistemas criptográficos con el fin de encontrar debilidades en los sistemas y romper su seguridad. El criptoanálisis es un componente importante del proceso de creación de criptosistemas sólidos. Gracias al criptoanálisis, podemos comprender los criptosistemas y mejorarlos identificando los puntos débiles. Un criptoanalista puede ayudarnos a trabajar en el algoritmo para crear un código secreto más seguro y protegido. Resultado del criptoanálisis es la protección de la información crítica para que no sea interceptada, copiada, modificada o eliminada. Otras tareas de las que pueden ser responsables los criptoanalistas incluyen evaluar, analizar y localizar las debilidades en los sistemas y algoritmos de seguridad criptográfica.

```
este
es
un
mensaje
de
prueba
```



2. Desarrollo (Parte 1)

2.1. Detecta el cifrado utilizado por el informante

Para poder detectar el cifrado utilizado, trabajé con la información a la cual tenía acceso, es decir, la llave de cifrado y la cantidad de mensajes cifrados. Sin embargo, carecía de detalles adicionales como un Initialization Vector (IV) o un contador. Esto me llevó a las siguientes conclusiones para determinar el método de cifrado más probable.

Como ya mencioné, se observó la carencia de un IV, lo que me llevó a descartar modos de cifrado en bloque como Cipher Block Chaining (CBC), que dependen de un IV. De igual forma, la ausencia de un contador en los mensajes me hizo descartar la opción del modo Counter (CTR) de cifrado en flujo. Además, se debe destacar que la documentación de CryptoJS menciona DES y 3DES como cifrados simétricos comunes. Esto fortaleció la probabilidad de que el informante hubiera utilizado el algoritmo de cifrado 3DES en modo

2.2 Logra que el script solo se gatille en el sitio usado por el informante

ECB, ya que 3DES es un cifrado ampliamente reconocido en el campo de la criptografía.

Adicionalmente, se llevaron a cabo pruebas con las dos opciones finales, DES y 3DES. Estas pruebas resultaron en la exitosa decodificación de los mensajes utilizando 3DES. Este hallazgo confirmó que el algoritmo de cifrado correspondía a 3DES en modo ECB.

2.2. Logra que el script solo se gatille en el sitio usado por el informante

Para poder realizar esta parte y las posteriores usaremos la herramienta Tampermonkey, la cual corresponde a una extensión que será instalada según el navegador, que en este caso corresponde a Firefox y que permite a los usuarios agregar secuencias de comandos personalizadas (scripts de usuario) a las páginas web que se visitan.

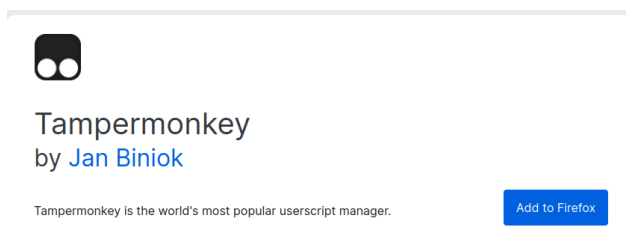


Figura 1: Instalación herramienta Tampermonkey

Para poder lograr que el script solo se gatille en el sitio usado por el informante, se debe configurar un parámetro del script, específicamente el campo **@match**, el cual corresponde a una directiva que se utiliza para especificar a qué páginas web o URLs se le aplicará el script. En este caso esa página corresponde a **https://cripto.tiiny.site/**.

```
// ==UserScript==
// @name         Lab4
// @namespace    http://tampermonkey.net/
// @version      0.1
// @description   Plugin Lab 4
// @author       Tobias Guerrero Cheuquepán
// @match        https://cripto.tiiny.site/
// @icon         https://www.google.com/sz/favicons?sz=64&domain=tiiny.site
// @grant        none
// @require      https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/crypto
// ==/UserScript==
```

Figura 2: Configuración campo @match

2.3. Define función que obtiene automáticamente el password del documento

Para identificar la clave correspondiente al cifrado, se llevó a cabo un análisis que reveló su posible presencia en el cuerpo del documento HTML, específicamente dentro de las etiquetas de párrafo “< p >< /p >”. Esta clave se representaría como un string que se concatenaría

con los caracteres en mayúscula encontrados a lo largo del contenido. A continuación, se presenta la función utilizada para llevar a cabo este proceso.

```
1 // Parte 1: Obtener la clave
2 function obtenerClave() {
3     var paragraph = document.querySelector('p'); // Ajusta el selector
4     // seg n la estructura de la p gina
5     var clave;
6
7     if (paragraph) {
8         var text = paragraph.textContent;
9         var mayusculas = text.match(/[A-Z]/g);
10
11         if (mayusculas) {
12             clave = mayusculas.join('');
13             console.log("La llave es: " + clave);
14         } else {
15             console.log("No se encontraron may sculas en el texto.");
16         }
17     } else {
18         console.log("No se encontr ning n elemento <p> con el texto.");
19     }
20
21     return clave;
22 }
```

Listing 1: Función para obtener la llave

2.4. Muestra la llave por consola

Después de compilar el script, se procede a inspeccionar los elementos de la página y a revisar la consola. De esta manera, la llave obtenida a través de la consola es la siguiente:



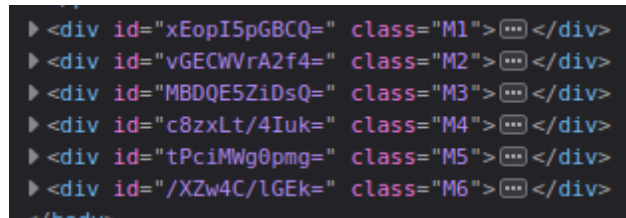
Figura 3: Llave obtenida

Por lo tanto la llave utilizada en el cifrado corresponde a “SEGUROSEGUROSEGURO-SEGURO”.

3. Desarrollo (Parte 2)

3.1. Reconoce automáticamente la cantidad de mensajes cifrados

Después de sumergirse en los detalles de la página, que fueron obtenidos a través de la inspección de elementos, al examinar el HTML en el cuerpo de la página, se identificaron 6 mensajes cifrados contenidos en los elementos `<div>`. A continuación, se presentan estos mensajes:



```

▶ <div id="xEopISpGBCQ=" class="M1">...</div>
▶ <div id="vGECWvRA2f4=" class="M2">...</div>
▶ <div id="MBDQE5ZiDsQ=" class="M3">...</div>
▶ <div id="c8zxLt/4Iuk=" class="M4">...</div>
▶ <div id="tPciMwg0pmg=" class="M5">...</div>
▶ <div id="/XZw4C/lGEk=" class="M6">...</div>

```

Figura 4: Mensajes cifrados

Para determinar la cantidad de mensajes cifrados mediante el script, se emplea una función que escanea elementos HTML `<div>` y evalúa si poseen un atributo `id` que sigue un patrón específico, correspondiente al formato de base64. Los elementos que cumplen con esta condición se incluyen en un arreglo denominado `mensajesCifrados`. Posteriormente, se muestra en la consola la cantidad de mensajes cifrados detectados.

```

1 // Parte 2: Contar los mensajes cifrados
2 function contarMensajesCifrados() {
3     var divElements = document.querySelectorAll('div'); // Ajusta el
4     selector seg n la estructura de la p gina
5     var mensajesCifrados = [];
6
7     divElements.forEach(function(divElement) {
8         var id = divElement.getAttribute('id');
9         if (id && /^[A-Za-z0-9+/=]+$/.test(id.trim())) {
10             mensajesCifrados.push(divElement);
11         }
12     });
13
14     console.log("Los mensajes cifrados son: " + mensajesCifrados.length);
15
16     return mensajesCifrados;
17 }

```

Listing 2: Función para obtener la cantidad de mensajes cifrados

3.2. Muestra la cantidad de mensajes por consola

Después de compilar el script, se procede a inspeccionar los elementos de la página y a revisar la consola. De esta manera, la cantidad de mensajes obtenidos corresponde a 6.

Los mensajes cifrados son: 6

Lab4.user.js:49:13

Figura 5: Número de mensajes cifrados

4. Desarrollo (Parte 3)

4.1. Importa la librería cryptoJS

Para importar la biblioteca CryptoJS, se accede al sitio web <https://cdnjs.com/> para obtener la URL necesaria. Esta URL se proporciona en el campo **@require** del script y corresponde a <https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/crypto-js.min.js>, como se obtiene de la página mencionada.

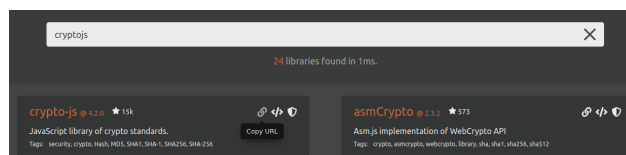


Figura 6: Extrayendo url

```
// ==UserScript==
// @name         Lab4
// @namespace    http://tampermonkey.net/
// @version      0.1
// @description   Plugin Lab 4
// @author       Tobias Guerrero Cheuquepán
// @match        https://crypto.tiiny.site/
// @icon         https://www.google.com/s2/favicons?sz=64&domain=tiiny.site
// @grant        none
// @require      https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.2.0/crypto-js.min.js#sha512-a+SU
// ==/UserScript==
```

Figura 7: Modificando campo @require

4.2. Utiliza SRI en la librería CryptoJS

El Subresource Integrity (SRI) es una característica de seguridad que se utiliza para verificar la integridad de los recursos externos, como archivos JavaScript, CSS o fuentes de fuentes web, que se cargan en una página web desde fuentes externas. El valor hash que se agrega al atributo integrity en la etiqueta script se utiliza para garantizar que el recurso externo no haya sido modificado por un atacante antes de que se cargue en el navegador del usuario.

La utilidad principal del SRI es proteger a los usuarios y las aplicaciones web contra ataques de Inyección de Código Malicioso o ataques Man-in-the-Middle. Estos ataques podrían implicar la modificación de los recursos externos que se cargan en una página web para incluir código malicioso o realizar cambios no deseados. Al utilizar SRI, el navegador realiza una verificación antes de cargar el recurso externo, comparando el valor hash del recurso descargado con el valor hash especificado en el atributo integrity. Si los valores no coinciden, el navegador bloqueará la carga del recurso, lo que protege la integridad de la página.

A continuación, se describe el proceso de extracción de este valor hash, que posteriormente se agregará en el campo previamente mencionado, **@require**.

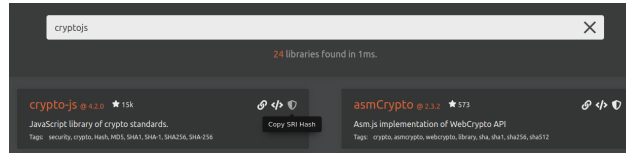


Figura 8: Extrayendo hash

Una vez obtenido, se concatena con la URL mencionada anteriormente, colocando un `&` entre esta y el hash.

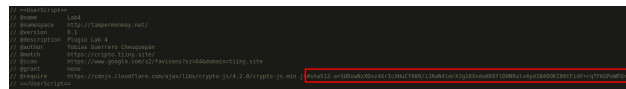


Figura 9: Modificando campo @require

4.3. Logra descifrar uno de los mensajes

Para llevar a cabo el proceso de descifrado, utilizaré la biblioteca CryptoJS. Usaré la llave previamente encontrada, junto con la cantidad de mensajes obtenidos, y aplicaré el algoritmo 3DES para realizar la tarea.

```

1 // Variable de estado para asegurarse de que el proceso solo se ejecute
  una vez
2 var procesoEjecutado = false;
3
4 // Parte 3: Descifrar y mostrar mensajes
5 function descifrarPrimerMensaje(mensajesCifrados, clave) {
6   if (procesoEjecutado) {
7     return; // Salir si el proceso ya se ha ejecutado
8   }
9
10  if (mensajesCifrados.length > 0) {
11    let desencriptar = CryptoJS.TripleDES.decrypt(mensajesCifrados[0].
      getAttribute('id'), CryptoJS.enc.Utf8.parse(clave), { mode: CryptoJS.
        mode.ECB }).toString(CryptoJS.enc.Utf8);
12    console.log(mensajesCifrados[0].getAttribute('id') + " " +
      desencriptar);
13
14    let resultadoElemento = document.createElement('p');
15    resultadoElemento.textContent = desencriptar;
16    document.body.appendChild(resultadoElemento);
17    procesoEjecutado = true; // Establecer la variable de estado a
      verdadero
18  }

```



```

19 }
20
21 // Ejecutar las partes en orden
22 var clave = obtenerClave();
23 var mensajesCifrados = contarMensajesCifrados();
24
25 if (clave && !procesoEjecutado) {
26     // Llamar a la funci n de descifrado del primer mensaje
27     descifrarPrimerMensaje(mensajesCifrados, clave);
28 } else {
29     console.log("No se pudo obtener la clave o no se encontraron mensajes
30     cifrados, o el proceso ya se ejecut .");
31 }

```

Listing 3: Función para descifrar un mensaje

Una vez corrido el script se obtiene lo siguiente:



Figura 10: Mensaje cifrado junto con su respectivo descifrado

4.4. Imprime todos los mensajes por consola

Para imprimir todos los mensajes en la consola, se realiza una modificación en la función mediante la incorporación de un bucle for, quedando de la siguiente manera:

```

1 // Parte 3: Descifrar y mostrar mensajes
2 function descifrarYMostrarMensajes(mensajesCifrados, clave) {
3     for (let i = 0; i < mensajesCifrados.length; i++) {
4         let desenscriptar = CryptoJS.TripleDES.decrypt(mensajesCifrados[i].
5         getAttribute('id'), CryptoJS.enc.Utf8.parse(clave), { mode: CryptoJS.
6         mode.ECB }).toString(CryptoJS.enc.Utf8);
7         console.log(mensajesCifrados[i].getAttribute('id') + " " +
8         desenscriptar);
9
10        let resultadoElemento = document.createElement('p');
11        resultadoElemento.textContent = desenscriptar;
12        document.body.appendChild(resultadoElemento);
13    }
14 }
15
16 // Ejecutar las partes en orden
17 var clave = obtenerClave();
18 var mensajesCifrados = contarMensajesCifrados();
19
20 if (clave && mensajesCifrados.length > 0) {
21     // Cargar CryptoJS

```

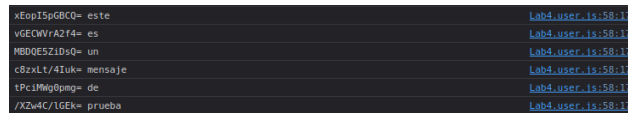
```

20 var script = document.createElement('script');
21 script.src = "https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/
crypto-js.min.js";
22 script.integrity = "sha384-S3wQ/100sbJoFeJC81UIr3J01x/0zNJpRt1bV+
yhpWQxPAahfpQtpxBsfn+Isslc";
23 script.crossOrigin = 'anonymous';
24
25 script.onload = function() {
26     descifrarYMostrarMensajes(mensajesCifrados, clave);
27 };
28
29 document.head.appendChild(script);
30 } else {
31     console.log("No se pudo obtener la clave o no se encontraron mensajes
cifrados.");
32 }

```

Listing 4: Función para descifrar los mensajes

Finalmente, al observar la terminal, se obtienen tanto los mensajes cifrados como los mensajes descifrados.



```

«Eep15p68C0» este Lab4_user.js:58:17
vGEQWvA2f4» es Lab4_user.js:58:17
MB0QESZ10sQ» un Lab4_user.js:58:17
c8zxLt/4Iuk» mensaje Lab4_user.js:58:17
tPc1Mw0pmg» de Lab4_user.js:58:17
/XZw4C/1GEK» prueba Lab4_user.js:58:17

```

Figura 11: Mensaje cifrado junto con su respectivo descifrado

4.5. Muestra los mensajes en texto plano en el sitio web

Del script anterior, se crea una etiqueta p en el body para así poder introducir el texto plano en el sitio web

```

1 let resultadoElemento = document.createElement('p');
2 resultadoElemento.textContent = desencryptar;
3 document.body.appendChild(resultadoElemento);

```

Listing 5: Código que agrega el texto plano al sitio web

4.6 El script logra funcionar con otro texto y otra cantidad de mensajes

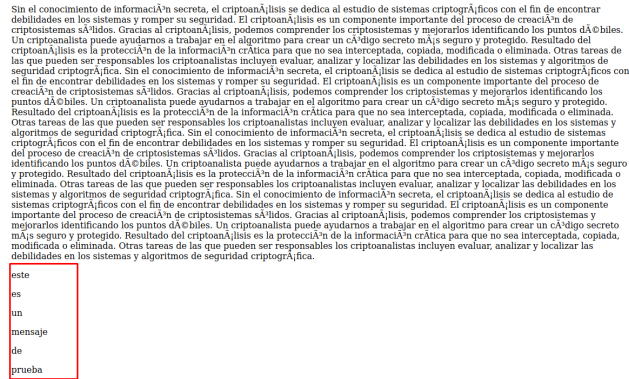


Figura 12: Mensajes en texto plano en la página web

4.6. El script logra funcionar con otro texto y otra cantidad de mensajes

Para poder demostrar que el script funciona con otro texto y otra cantidad de mensajes, se tuvo que borrar todo el HTML actual que existía en la página, generando uno nuevo siguiendo el mismo formato de tener una etiqueta `p` y contenedores `div` con los mensajes cifrados. En este caso, se enviaron 10 mensajes y la clave también cambió. Además, el string enviado fue “Descifrar este mensaje es como buscar agua en el desierto”. Para convertir cada palabra cifrada en Base64, utilicé la siguiente página <https://www.devglan.com/online-tools/triple-des-encrypt-decrypt>. A continuación, se detalla el código.

```
1 function createNewContent() {
2     // Mensajes cifrados en Base64
3     var mensajesCifrados = [
4         'lZz/tIfhiFagxWRP1R/4oA==',
5         '9qHjrcZ7sa8=',
6         'x1MzIoAB/fA=',
7         'jNZ7KNF9Jtg=',
8         'GzLqG+EZm2w=',
9         'FS1CR7RHUIA=',
10        'jOTR5/snS8g=',
11        'ZQGHrSk3b9E=',
12        'ZavsZkAx6fo=',
13        'OHLT8MLfLXEuZ1jCWOT6bQ==',
14    ];
15
16    // Nuevo contenido del mensaje en el p rrafo (p)
17    var newMessage = '
18    En un reino lejano, exist a un bosque m gico conocido como el
19    Bosque de las Maravillas. En ese lugar, una antigua profec a hablaba
20    de un valiente aventurero llamado Max Gordon con el poder de descubrir
21    los tesoros ocultos.'
22 }
```

4.6 El script logra funcionar con otro texto y otra cantidad de mensajes

```
20      Un d a , un joven llamado Juan junto a sus amigos perros Bob, Tom
    y Kyle decidi emprender un viaje hacia el Bosque de las Maravillas.
    Con su mochila llena de sue os y determinaci n , se adentr en el
    bosque en busca de aventuras.

21
22      A medida que exploraba, Juan se encontr con criaturas m gicas ,
    como el majestuoso Drag n de las Nubes y el amigable Hada del Bosque.
    Estas criaturas le otorgaron regalos especiales que lo ayudaron en su
    b squeda .

23
24      Despu s de muchas semanas de exploraci n , Juan finalmente lleg
    al coraz n del bosque, donde descubri una puerta antigua que lo
    llev a un mundo secreto lleno de tesoros invaluable. En ese lugar,
    cumpli la profec a y se convirti en el legendario "Guardi n de
    los Tesoros".

25
26      La historia de Juan se transmiti de generaci n en generaci n ,
    recordando a todos que, con valent a y determinaci n , se pueden
    descubrir tesoros ocultos en los lugares m s inesperados.

27      `;

28
29      // Lista de clases para los contenedores div (M1, M2, ...)
    var divClasses = ["M1", "M2", "M3", "M4", "M5", "M6", "M7", "M8",
30      "M9", "M10"];

31
32      // Eliminar todo el contenido HTML existente en la p gina
    document.head.innerHTML = "";
33      document.body.innerHTML = "";

34
35
36      // Crear un nuevo elemento HTML para el mensaje en el p rrafo (p)
    var newParagraph = document.createElement("p");
37      newParagraph.textContent = newMessage;
38      document.body.appendChild(newParagraph);

39
40
41      // Agregar mensajes cifrados en divs al cuerpo de la p gina
    for (let i = 0; i < mensajesCifrados.length; i++) {
42          var nuevoDiv = document.createElement('div');
43          nuevoDiv.className = divClasses[i];
44          nuevoDiv.setAttribute('id', mensajesCifrados[i]);
45          document.body.appendChild(nuevoDiv);
46      }
47  }
48  }

49
50      // Llamar a la funci n para actualizar el contenido
51      createNewContent();
```

Listing 6: Código que crea el nuevo HTML

4.7 Indica url al código .js implementado para su validación DESARROLLO (PARTE 3)

Luego se obtiene lo siguiente a partir de lo anterior.

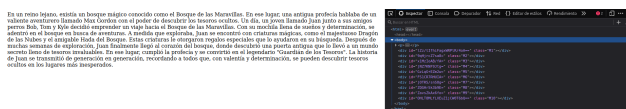


Figura 13: Prueba con nuevo texto y nueva cantidad de mensajes

Finalmente, una vez agregadas las mismas funciones de las partes previas, se obtiene el resultado esperado, es decir, la llave, la cantidad de mensajes y el mensaje en claro.

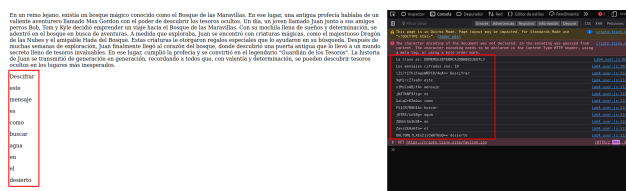


Figura 14: Resultados con el nuevo texto

4.7. Indica url al código .js implementado para su validación

Los scripts que se utilizaron están disponibles en el siguiente repositorio: <https://github.com/tobiasker19/Labs-Cryptography>

Conclusiones y comentarios

Este informe detalla el proceso de creación de un script utilizando la extensión Tampermonkey para llevar a cabo la descifración de mensajes en una página web. Se identificó el cifrado 3DES como el método empleado en la página web de referencia, y se desarrolló un script capaz de extraer la clave de cifrado oculta en el contenido de la página. Además, el script tiene la capacidad de identificar automáticamente la cantidad de mensajes cifrados presentes en la página y procede a descifrarlos utilizando la clave recuperada. Para garantizar la seguridad y la integridad del proceso, se implementó la biblioteca CryptoJS y se aplicó la técnica SRI. El resultado final es un script plenamente funcional que descifra y muestra los mensajes cifrados en la página web, subrayando la importancia de mantener la seguridad de los datos y salvaguardar las claves de cifrado. Finalmente, este laboratorio pone de manifiesto cómo las herramientas y técnicas adecuadas pueden ser utilizadas para abordar desafíos de seguridad en sitios web, resaltando la relevancia de comprender el funcionamiento de los algoritmos de cifrado y su aplicación práctica.